Depth-First Search

- For each unexplored \( u \in V \)
  - \( \text{DFS}(G, u) \)
- \( \text{DFS}(\text{graph } G, u \in V) \)
  - mark \( u \) as explored
  - for each \( \{u, v\} \in E \)
    - if \( v \) is unexplored then \( \text{DFS}(G, v) \)

Running time \( O(m + n) \)

Graph Search Problems

- Given a graph \( G \)
  - Check if there is a path between two given vertices \( s \) and \( t \)
  - Decide if \( G \) is connected
  - Identify the connected components of \( G \)
- All these problems can be solved directly using any kind of vertex traversal, including DFS
**TOPOLOGICAL SORTING**

- A topological order of a directed graph $G$ is a bijection $f : V \rightarrow \{1, ..., n\}$ such that if $(u, v) \in E$ then $f(u) < f(v)$

- An undirected graph is a clique iff for all distinct $u, v \in V$, $(u, v) \in E$

- Poll 1: Which of the following undirected graphs can have an orientation that does not admit a topological sorting?
  1. Tree
  2. Clique
  3. Both
  4. Neither
**Topological Sorting**

- Clearly if a graph has a cycle then it does not have a topological order.
- We will give an algorithm that finds a topological order given any directed acyclic graph.
- A sink vertex is a vertex with no outgoing edges.
- *Lemma:* Every directed acyclic graph has a sink vertex.

**Proof of Lemma**

- Suppose for contradiction that every vertex has an outgoing edge.
- By following the outgoing edges, after at most $n$ steps we must revisit an vertex we’ve already seen, leading to a cycle! ■

**Naïve Algorithm**

- $p \leftarrow n$
- while $p \geq 1$
  - If the graph doesn’t have a sink then return “not acyclic”
  - else find a sink $v$ and remove it from $G$
  - $f(v) \leftarrow p$
  - $p \leftarrow p - 1$

*Running time?*
**Better Algorithm Via DFS**

- \( p \leftarrow n \)
- *For each* unexplored \( u \in V \), DFS\((G, u)\)

DFS\((\text{graph } G, u \in V)\)

- mark \( u \) as explored
- *for each* \( \{u, v\} \in E \), *if* \( v \) is unexplored *then* DFS\((G, v)\)
- \( f(u) \leftarrow p \)
- \( p \leftarrow p - 1 \)

**Correctness**

- Theorem: If \( G \) is acyclic and \( (u, v) \in E \) then \( f(u) < f(v) \)
- Proof: We consider two cases
  - Case 1: \( u \) is discovered before \( v \), then because \( (u, v) \in E \), \( v \) will be explored before DFS\((G, u)\) returns
  - Case 2: \( v \) is discovered before \( u \), then we cannot discover \( u \) from DFS\((G, v)\) because that would imply a cycle, so DFS\((G, u)\) is run after DFS\((G, v)\) terminates

**Weighted Graphs**

- It is often useful to consider graphs with
  - weights
  - lengths
  - distances
  - costs
associates to their edges
- Model as a cost function

\( c : E \rightarrow \mathbb{R}^+ \)
**Minimum Spanning Tree**

The year: 1926
The place: Brno, Moravia
Our hero: Otakar Borůvka

Borůvka’s had a pal called Jiříček Saxel who worked for Západomoravské elektrárny (the West Moravian Power Plant company). Saxel asked him how to figure out the most efficient way to electrify southwest Moravia.

**Minimum Spanning Tree**

* MST problem:
  - Input: Graph $G$, cost function $c: E \to \mathbb{R}^*$
  - Output: $E' \subseteq E$ such that $(V, E')$ is connected and $\sum_{e \in E'} c(e)$ is minimized

* Example: The MST has cost 42
Obviously the optimal solution forms a tree!

**Number of MSTs**

- **Assumption** (for convenience): Edges have distinct weights
- **Poll 2:** What is the max \#MSTs that a 3-clique can have?
  1. 1
  2. 2
  3. 3
  4. 4

Under the assumption, the MST is unique! This will follow as a corollary from the next proof.
**PRIM’S ALGORITHM**

• $V’ \leftarrow \text{arbitrary } \{u\}, E’ \leftarrow \emptyset$
• While $V’ \neq V$
  • Let $(u, v)$ be a minimum cost edge such that $u \in V’$, $v \notin V’$
  • $E’ \leftarrow E \cup \{u, v\}$
  • $V’ \leftarrow V’ \cup \{v\}$

**PROOF OF CORRECTNESS**

• Fix an MST $T$; we will show that for every $0 \leq k \leq n$, the first $k$ edges added by the alg are in $T$
• The proof is by induction
• Base case ($k = 0$) is vacuously true
• Induction step: Suppose the algorithm has added $k$ edges so far that are in the MST; show that next edge is also in the MST

Running time? It’s clearly polynomial, and that’s surprising!
**Proof of Correctness**

- Consider the current $V'$
- Let $e = \{a, b\}$ be the next edge added by the alg
- Suppose $e$ is not in the MST $T$ (shown in red)
- $T$ has a path $a \to b$
- Let $e' = (c, d)$ be the first edge on the path that exits $V'$

---

**Proof of Correctness**

- Consider $T' = T \cup \{e\} \setminus \{e'\}$
  - Its cost is lower than $T$
  - It has $n - 1$ edges
- $T'$ is connected because
  - any path $u \to c \to d \to v$ that uses $e'$ is replaced by $u \to c \to a \to b \to d \to v$
- So $T$ is not an MST! $\blacksquare$

---

Why does the proof imply that the MST is unique?
THE MST CUT PROPERTY

- A similar proof shows: Let $G$ and $V' \subseteq V$, and let $e$ be the cheapest edge between $V'$ and $V \setminus V'$, then $e$ is in the MST.
- Using this it is not hard to show that any natural greedy algorithm works, e.g.,
- Kruskal’s Algorithm:
  - Go through edges from cheapest to most expensive
  - Add the next edge if it doesn’t create a cycle

RUN-TIME RACE FOR MST

- A naïve implementation of Kruskal and Prim runs in time $O(m^2)$
- A better implementation runs in time $O(m \log m)$
- That’s very good!
- In practice, these algorithms are great
- Nevertheless, algorithms and data structures wizards tried to do better
**Run Time Race for MST**

1984: Fredman and Tarjan invent the Fibonacci heap data structure

\[ O(m \log m) \rightarrow O(m \log^* m) \]

---

**Run Time Race for MST**

1986: Gabow, Galil, Spencer, and Tarjan improved the algorithm

\[ O(m \log^* m) \rightarrow O(m \log \log^* m) \]

---

**Run Time Race for MST**

2000: Chazelle invents the soft heap data structure

\[ O(m \log \log^* m) \rightarrow O(m \cdot \alpha(m)) \]

What is \( \alpha(\cdot) \)?
**Detour: \( \alpha(\cdot) \)**

- \( \log^*(n) = \# \text{times you need to do } \log \text{ to get down to } 1 \)
- \( \log^*(m) = \# \text{times you need to do } \log \text{ to get down to } 1 \)
- \( \log^{**}(m) = \# \text{times you need to do } \log^* \text{ to get down to } 1 \)
- ...  
- \( \alpha(m) = \# \text{stars you need to do so that } \log^{**}(m) \leq 2 \)

It is incomprehensibly slow growing!

**Run Time Race for MST**

- 2002: Pettie and Ramachandran give an optimal MST algorithm
- But... nobody knows what its running time is!

**Summary**

- Terminology:
  - Topological order
  - Weighted graph
  - Minimum spanning tree
- Algorithms:
  - DFS
  - Topological sort via DFS
  - Prim’s Algorithm
- Theorems:
  - MST Cut property